06

Diferencias entre ArrayList y LinkedList

Transcripción

[00:00] Hola. ¿Cómo están? Vamos a iniciar nuestra clase 8. Nuestra clase 8 va a ser un poco diferente porque vamos a ver las diferencias entre LinkedList y ArrayList. LinkedList y ArrayList son muy parecidos, pero tienen una peculiaridad para cada método, que puede ser nuestro método add, get y remove, que son los métodos principales. Para esto, vamos a verlo en la práctica.

[00:28] Entonces vamos a colocar nuestra Clase7, vamos a colocar aquí Clase8. Vamos a remover todo lo que es cursos y vamos a hacer lo siguiente. Vamos a crear nuestras dos listas, final List y vamos a colocar una lista de Integer. Vamos a colocar listas = new ArrayList. Aquí digitamos List>(), abrimos y cerramos paréntesis, cerramos llaves.

[01:26] Dentro, abrimos y cerramos llaves también para hacer nuestro método add. Add va a tener dos, uno llamado, Linked list, de tipo también entero, Integer, y otro add que sería el new ArrayList. Que sería también Integer. Perfecto.

[02:05] Aquí dice que no es necesario, porque aquí estamos diciendo que va a ser entero, entonces no es obligatorio colocar entero, colocar Integer. Ahora hacemos lo siguiente. Aquí creamos el método for que lea nuestra lista, crea aquí List lista : listas. ¿Aquí qué sucede? Aquí está diciendo que no está estático.

[02:45] ¿Entonces aquí tenemos que colocar que cosa? Static, vamos a colocar aquí también public static final. Listo. Métodos estáticos tienen que trabajar siempre con atributos estáticos, que estén fuera de él. Una vez hecho esto, utilizamos aquí el for y hacemos lo siguiente: final String, vamos a colocar el nombre de la implementación, que sería lista.getClass().getSimpleName() que sería el nombre de nuestra ArrayList o LinkedList, que sería el nombre de nuestra clase.

[03:37] Esto de aquí, imprimiría en string. Vamos a ver nuestros métodos que vamos a comparar: Método add, método get, método remove. El método add, tenemos un long inicio = System.currentTimeMillis(); que sería el tiempo en milisegundos de ahora, cuando va a ser ejecutado.

[04:09] Digitamos un for simple, solamente que nuestro for va a tener un tamaño máximo de 10.000.000. 10.000.000 de registros vamos a crear. Ahí colocamos i++. Una vez hecho esto, colocamos aquí lista.add(i); tenemos después el fin, y aquí también System.currentTimeMillis();.

[04:46] Una vez hecho esto tenemos aquí el long, la duración. Vamos a ver cuánto fue lo que duró, que sería fin - inicio. Una vez hecho esto, vamos a imprimir System.out.println que sería aquí el nombre de nuestra implementación. Vamos a concatenar. Queremos el método add, que es lo que estamos comparando primero. Colocamos add y colocamos la duración.

[05:28] Una vez hecho esto, tenemos ya nuestra primera parte. Después, ahora vamos a utilizar el get, este de aquí que sería Inicio = System.currentTimeMillis(); Después hacemos otro for simple, y en nuestro for vamos a utilizar la misma cantidad de registros, que serían, ¿Cuántos? 10.000.000.

[06:09] Una vez hecho esto, vamos a utilizar lista.get(i); entonces vamos a hacer un get de todos nuestros objetos ya registrados anteriormente y vamos a estar

haciendo gets constantes. Cuando termine de hacer todos los gets, tenemos un fin.

[06:34] Dice fin = System.currentTimeMillis(); también y aquí sería una duración igual a fin - inicio. Duración. Una vez hecho eso, imprimimos de nuevo y utilizamos ¿quién lo va a crear? Get. Para el delete o remove vamos a hacer lo mismo. Solamente que en lugar de get, utilizamos remove. Y en lugar de get, aquí también remove. Ahora ejecutamos.

[07:33] Aquí va a demorar un poco porque está recorriendo toda la lista de 10.000.000. Ahora está haciendo todos los gets. Vamos a ver si termina, si no reducimos el tiempo. Primero está ejecutando, ¿qué cosa? El LinkedList. Entonces el LinkedList para get no está siendo tan performático. Se está demorando mucho.

[08:26] Vamos a hacer lo siguiente, vamos a dar un stop. Vamos a dar aquí 100.000 registros. Aquí está, aquí dice lo siguiente, vamos a ver aquí. Todos están con 100, 100, 100. Aquí está el add, el get y en el remove sucedió un error. ¿Cuál fue el error?

[09:43] El error es que en el LinkedList, ahí lo vamos a explicar aquí con más detalle, pero para el LinkedList necesitamos hacer lo siguiente. En lugar de 100.000, necesitamos colocar 99999, que sería desde mayor a menor, para hacer el remove. Entonces aquí cambiaría nuestro contexto, que sería mayor o igual a 0 y nuestro i sería --. Ahora ejecutemos.

[10:23] 15 milisegundos demoró en hacer el add. El get demoró esta cantidad de registros. El remove demoró 4. Prácticamente el remove lo está haciendo casi igual con ArrayList, el add lo está haciendo bien rápido y el get, aquí estuvimos viendo, lo hizo mucho más rápido. ¿Entonces qué tenemos aquí? El ArrayList, el get conseguimos hacer de una forma mucho más eficiente que LinkedList.

[11:03] El remove puede ser casi parecido. El add tuvo una cierta limitación en LinkedList, pero ahora viene lo siguiente. Cuando tenemos, vamos a comentar aquí, un LinkedList, la ventaja es que nosotros podemos adicionar en cualquier lugar de LinkedList, en cualquier posición y lo va a hacer de una forma más rápida que un ArrayList. Por ejemplo, en LinkedList, cuando crea una lista, va creando tipo una relación entre ellas.

[11:51] ¿Qué es lo que hace? Por ejemplo, queremos adicionar en esta posición, el LinkedList automáticamente, ¿qué hace? Quita esta lista de aquí, quita esto de aquí y lo entrelaza. Entonces aquí tendríamos B1 por ejemplo, y automáticamente tendría aquí el B2. Muy diferente al ArrayList. En ArrayList nosotros vamos tipo de un sentido. ¿Qué sucede?

[12:36] Cuando hacemos el ArrayList un add por ejemplo, ¿qué vamos a hacer? El ArrayList, si queremos en la posición 2, ¿qué va a hacer? Esta posición sería 0 y la posición 1. Pero sería el número 2. ¿Qué hacemos? El ArrayList hace aquí lo siguiente: remueve esto de aquí, adiciona entonces sería 3, 4 y5 y aquí colocaría el B1.

[13:08] Entonces, si nosotros tenemos una lista de 1.000.000 de registros, por ejemplo, imaginemos que los quiero adicionar en la posición 3. Él va a empujar desde la posición 4 hasta la posición 1.000.000, va a empujar toda la lista +1, +1, +1 y va a demorar bastante. Pero en el LinkedList, lo que él hace es simplemente, quita la relación de una, adiciona con otro y listo. Es mucho más eficiente.

[13:33] Pero para hacer los gets, es menos eficiente. Y el remove prácticamente sigue siendo casi igual en los 2. Entonces cada uno tiene sus ventajas, pero por eso también estuvimos viendo utilizar las interfaces. ¿Por qué? Porque imaginémonos que ya tenemos una lista, ya tenemos el sistema y utilizamos ArrayList y vemos que está demorando bastante. ¿Qué hacemos? Vamos a cambiarlo para el LinkedList.

[14:01] La parte de adicionar las posiciones está demorando bastante, entonces cambiamos para LinkedList. Y automáticamente toda nuestra lista se convirtió en LinkedList. Por eso es una gran ventaja trabajar prácticamente con nuestra interface lista y todas las implementaciones que usemos, las podemos cambiar de aquí a un tiempo.

[14:20] Esta sería básicamente nuestra clase número 8 y estaríamos terminando el módulo de relacionando listas de objetos. Nos vemos en la siguiente clase. Muchas gracias.